Generation and Validation of Workflows for On-demand Mapping

Nick Gould and Omair Chaudhry
Manchester Metropolitan University
Manchester, UK
emails: {nicholas.m.gould@stu.mmu.ac.uk, o.chaudhry@mmu.ac.uk}

***Abstract*— The paper presents a method to automatically select and sequence the tasks required to build maps according to user requirements. Workflows generated are analysed using Petri nets to assess their validity before execution. Although further work is required to select the optimal method for generating the workflow and to execute the workflow, the proposed method can be used on any workflow to assess its validity.**

Keywords-automated map generalisation; workflows; Internet mapping; Petri nets.

I. INTRODUCTION

The development of Google Maps and similar products has led to a vast number of 'mashups' where users can overlay their own data on Google Maps backgrounds and make the resultant map available to others. The problem with this approach is that the user is limited to the background maps supplied by Google; there is no, or very little, flexibility to vary the content depending on the context and there is no data integration [1]. This is highlighted in Fig. 1 where the street names are obscured by overlaid cycle routes. Further problems may occur when the scale changes. For instance, a minor road that may be part of a cycle route may disappear at smaller scales since the two datasets are independent.

What is required is a system to allow data from a variety of sources to be mapped at a variety of scales. Since, the possible combination of datasets and scales is too numerous to be pre-defined, on-demand generalisation (deriving smaller scale maps from larger scale maps) is necessary.

Cartographic generalisation is a complex process [2] and much effort has gone in to developing automation techniques that reduce or eliminate human involvement [3].

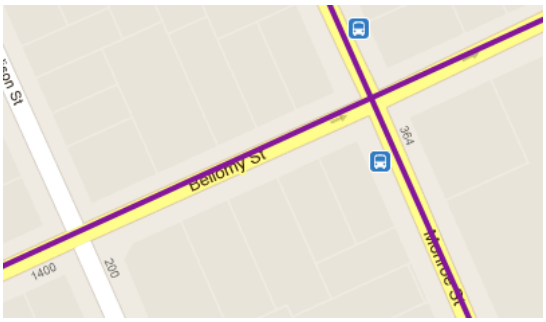

Figure 1. Google Maps with cycle routes overlaid

The focus has, until recently, been on allowing National Mapping Agencies (NMAs) to automate the production of maps at different scales from a single master source [4][5]. Automatic generalisation is applied to a pre-defined set of map features at pre-defined scales to produce a pre-defined set of products. However, the advance of neo-geography and Volunteered Geographic Information [6] means that on-demand generalisation is required allowing users to integrate their data with that of NMAs and other mapping resources. There have been attempts to generate online on-demand maps to user requirements, but such systems have been developed by applying a fixed sequence of generalisation operations to known datasets [7][8].

An on-demand mapping system will require a number of components including a means of taking high level user requirements (e.g., "I want a city-wide map of road accidents") and producing a machine-readable specification of the map [9]. The system will also need a knowledge base to store cartographic rules of the type: "if the scale is greater than 1:30,000 omit minor roads". A set of map generalisation services are then required to satisfy such rules or constraints. Traditionally the selection of map generalisation operators and their sequencing is done by cartographic experts, but for on-demand mapping, aimed at the non-expert user, a system is required that can automatically generate, validate, execute and monitor these operations; in other words a workflow needs to be generated and executed [9]. The focus of this research is on developing a workflow engine that, given the specification, using the rules, will automatically select, sequence, and execute the map generalisation services required to generate the map or spatial output.

This paper describes the initial attempts to automatically generate a workflow for building a map based on user requirements and suggests how to validate that workflow.

To illustrate the process, a use case involving the mapping of road accidents will be employed. Fig. 2 represents a detailed map of accidents at a road junction.

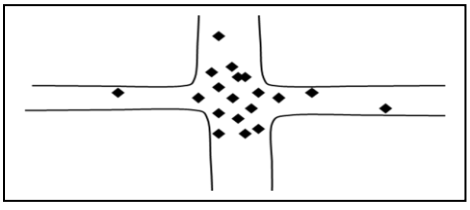

Figure 2. Accidents at a road junction

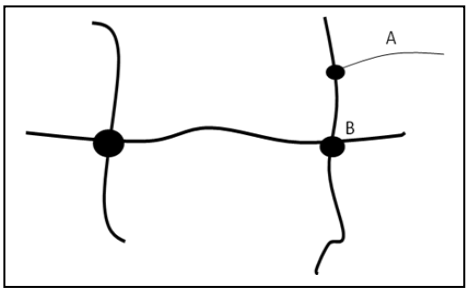

Figure 3. Generalised data at a small scale

To represent all of the data at a smaller scale the road network is generalised by *eliminating* any minor roads and *collapsing* (reducing to single lines) the major roads. To avoid information overload the accidents are *clustered* (Fig. 3). Elimination, collapse and clustering are common generalisation operations. The junction in Fig. 2 is represented by 'B'.

The generalised map serves to highlight accident hot spots. However, by removing the minor road 'A', context is lost, since the cluster will appear to be on a straight section of road, so a step is required to reinstate those minor roads that intersect a cluster. What we have is a set of tasks that have to be carried out, some of which are in a particular order, i.e., we cannot reinstate the minor roads until we have created the clusters. Since there a number of tasks to execute, some of which have to be completed before others can start, a workflow is required to express these relationships. In addition that workflow has to be valid, i.e., all of the tasks must, at least, be called.

The method used was based on the premise that workflow definitions can be analysed using Petri nets for flaws that would stop the workflow from completing execution [10][11].

Firstly, a technique was implemented to generate the list of tasks and their dependencies based on applying user requirements to a set of applicable rules (described in Section II). From this a workflow definition could be created, represented by a directed graph (Section III). A method was developed to produce a Petri net from a given directed graph (Section IV). The Petri net could then be analysed for flaws in the workflow definition (Section V).

II. GENERATING A LIST OF TASKS AND DEPENDENCIES

There are a number of different techniques for automatically generating workflows including Case Based Reasoning [12] and product structures, where the map is treated as a product that has to be constructed from component parts [13].

The technique used in this research was one that employs user preferences to define the selection and execution of a set of rules [14]. This technique was selected because of its focus on the user's needs, which is essential for on-demand mapping.

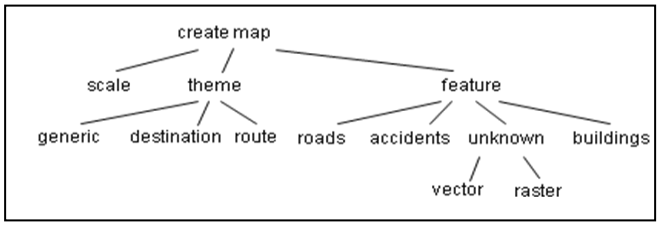

Figure 4. Knowledge hierarchy

The user preferences are gathered by navigating a knowledge/rule hierarchy (Fig. 4). If a particular branch is not selected by the user than that branch is closed off. For example, if the user does not select an 'unknown' feature type they are not prompted for 'vector' or 'raster'. In the prototype the user is simply prompted for his or her preferences using text boxes and drop down boxes in a web page. Each leaf node in the hierarchy has one or more associated rules; these are added to a set of applicable rules as the user requirements are gathered.

If the user selects the 'roads' feature type then the rules associated with that feature type (R1, R2, R3) are added to the set of applicable rules. Rules consist of a condition and an action (e.g., *insert* a task to the workflow or *order* two tasks) and are stored in an XML file (Fig. 5). Using XML allows for the use of schemas to enforce correct structure.

The gathered user requirements are held in memory as ordered pairs, for example:

< scale,50000 >
< theme,generic >
< featureType,roads >
< featureType,accidents >

```
...
<featureType name="roads">
    <rules>
      <rule id="R1">
       <condition>scale >= 5000 AND
featureType = roads</condition>
       <action>insert(t1)</action>
      </rule>
      <rule id="R2">
        <condition>scale >= 5000 AND
featureType = roads</condition>
        <action>insert(t2)</action>
      </rule>
      <rule id="R3">
      <condition>scale >= 5000 AND featureType
= roads</condition>
      <action>order(t2,t1)</action>
      </rule>
    </rules>
</featureType>
...
```

Figure 5. Knowledge/rule hierarchy (partial) as XML

Once these have been collected, the applicable rules are then evaluated, checking rule conditions against user preferences to generate a set of tasks and a set of task dependencies. For example, if the user has selected the feature type "roads" and a map scale of greater than 1:5000 then the conditions for rules R1, R2 and R3 (Fig. 5) will be met and their actions triggered, e.g. task t1 is inserted into the workflow. The action 'order(t2,t1)' means task t1 is a dependency of task t2 and should only be run after t2 has been executed.

Using the above use case, the selected tasks may be:

t1: collapse roads
t2: delete minor roads
t9: add copyright notice for roads dataset
t3: cluster accidents
t8: reinstate minor roads on clusters

and the dependencies:

$t2 \prec t1$
$t1 \prec t8$
$t3 \prec t8$

In this case, there are five tasks to perform and there are three dependencies (or precedence constraints). For example, we want to delete any minor roads (task t2) before we collapse the roads (task t1) as it is inefficient to collapse a subset of the road network that we are later going to delete. Task t9 is not involved in any dependency and is classified as an independent task.

The above output can be expressed as a directed graph where tasks are represented as nodes and dependencies as edges (Fig. 6).

However, the graph does not constitute a workflow. The next section describes why and then what is needed to construct a workflow definition.

III. CREATING A WORKFLOW DEFINITION

The directed graph (Fig. 6) generated from the example set of task dependencies does not make up a workflow definition. This is because there is no place for any independent tasks (in our case task t9). In addition, the following rules must be satisfied for a workflow definition according to [15]:

- The workflow graph should have a single source node and a single sink node
- Every other node should have at least one parent and at least one child

This ensures that the workflow has a defined start and end and that there are no unnecessary tasks.

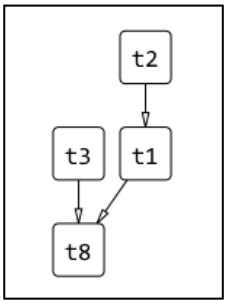

Figure 6. Directed graph based on dependencies

A workflow definition directed graph can be created by the following procedure:

1. Add start (A) and end (B) nodes
2. Create an edge for every dependency
3. For every node that has no children add an edge to the end node
4. For every node without a parent add an edge from the start node
5. For each independent tasks link the task directly to the start and end node.

The revised graph can be seen in Fig. 7.

The method so far has produced a workflow definition for the given case study but is it valid? For instance, it is relatively easy to ensure that there are no directly contradictory dependencies between the selected tasks so that both $t1 \prec t2$ and $t2 \prec t1$ did not appear in the same workflow. However, indirectly contradictory dependencies such as that seen in Fig. 8 would be harder to identify. In this example the dependency "t3 precedes t14" has introduced *deadlock* [15] into the workflow. Task t3 will not start until t8 starts; but t8 will not start until t14 starts, which will not start until t3 starts. So, tasks t14, t8, t5, t3 and subsequent tasks will never be executed.

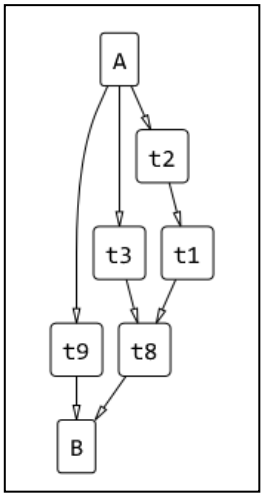

Figure 7. Workflow definition graph

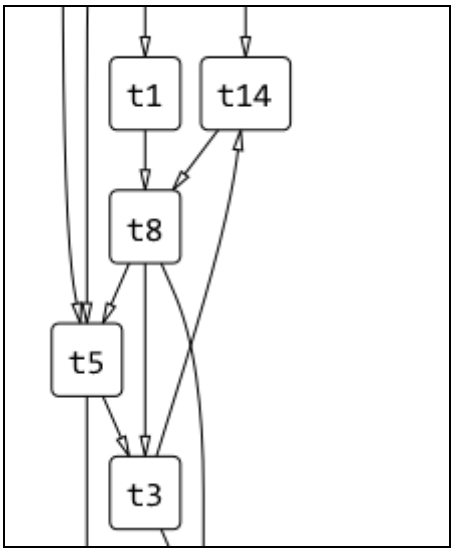

Figure 8. Deadlock in a workflow

A method of testing the soundness of a workflow before attempting execution is needed. The simplest way of checking for deadlock is by performing a topological sort on the graph. However, the application of Petri nets will allow for a more expressive form of graph and the ability to describe more complex workflow patterns [16] than that described above. In addition to describing workflows, the *mathematical foundations* of Petri nets [17] allow for the analysis of workflows and are applicable to more complex analysis than the deadlock problem [10][11]. Extensions to Petri nets, such as coloured Petri nets, which allow for the investigation of delays and throughput, have been defined formally [11]. Petri nets offer a number of advantages over PERT charts such as the ability to model nondeterministic behaviour and loops in the workflow [31]. The adoption of Petri nets at an early stage will allow the design to be scaled to more complex workflows. But, first, we need to generate the Petri net from the directed graph.

IV. Generating a Petri Net

A Petri net is a particular class of directed graph, defined as a bipartite directed graph consisting of two types of nodes called transitions and places [17]. Nodes are linked by arcs such that arcs cannot link a place to a place or a transition to a transition. Transitions, denoted by rectangles, represent events or, in our case, workflow tasks. Places denoted by circles, represent states (Fig. 9).

Staines [18] describes the process for generating a directed graph from a Petri net, which can be reversed to generate a Petri net. The procedure used is as follows:

1. Nodes (tasks) are converted to transitions
2. Each edge generates arc-place-arc
3. Extra places are added preceding the start node (A) and following the end node (B).

The Petri net generated from the workflow shown in Fig. 7 can be seen in Fig. 9.

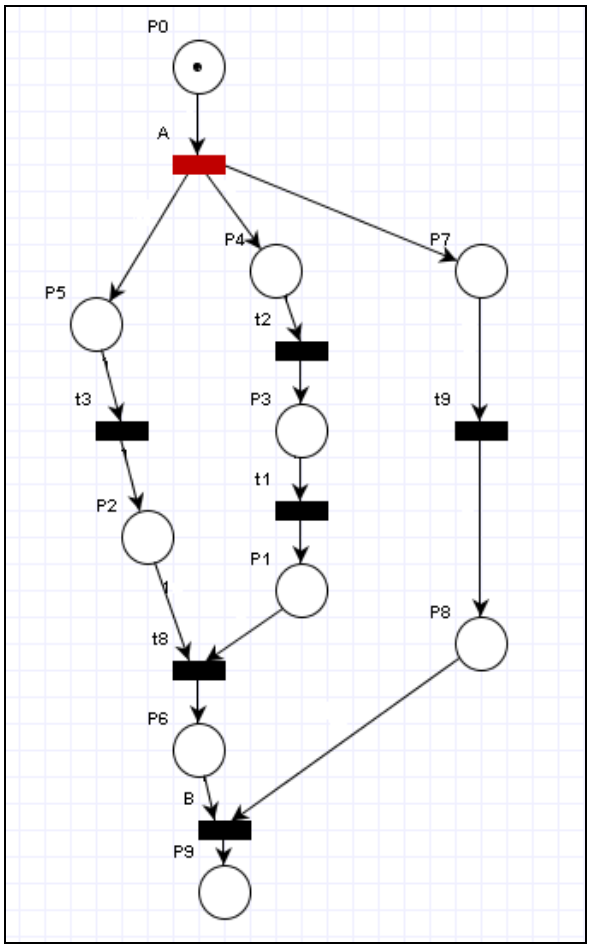

Figure 9. Petri net for valid workflow

The starting place, P0, contains a single *token*. Tokens can be used to model the workflow. A transition may be fired only if there are one or more tokens in all of its input places [17]. In this example, transition A can be fired. When a transition fires it takes a token from each of its input places and places a token in each of its output places. So after the firing of transition A, there will be a token in each of the places P5, P4, P7 (but no longer P0) thus enabling transitions t3, t2 and t9. Note that t8 will not be fired until both t3 and t1 have, which models the original dependencies.

Code was written to generate an XML file in a format that can be read by the PIPE software [19]. PIPE can then be used to visualise and animate the Petri net firings to ascertain whether the workflow is executable.

Deadlock can be identified visually or by using a Petri net analysis tool such as PIPE. It needs, however, to be identified as part of the on-demand mapping system. The following section describes how this was done.

V. Validating the Workflow

Our system generates a *workflow net* [15], a particular type of Petri net such that:

- The net has a single source and a single sink node

- Every tasks lies on a directed path between the source and the sink nodes.

However, as has been shown, a workflow net containing anomalies such as deadlocks can still be generated. A *sound* process is defined as one that contains no unnecessary tasks and where every case submitted to the process must be completed in full and no reference to it remaining in the process, i.e., for every token that is in the start place there is one token in the end place and no others in the net [15].

There is a number of, sometimes complex, techniques for checking the soundness of a workflow net. Fortunately the Petri nets derived from our workflow generation are a particular sub-class of Petri nets known as conflict free or T-systems where every place has no more than one input and one output transition [20]. In effect conflicts are ruled out; there are no logical ORs in the system. This makes them easier to analyse [21].

In the prototype the Petri net is checked for deadlock by looping through the set of places and firing any transitions that are enabled until no more transitions can be fired. If all the transitions are fired then the workflow is valid, if there are transitions that cannot be fired then these are listed.

In addition to the case study, the method was tested on a number of randomly generated task lists and dependencies. A demonstration version of the prototype can be seen at www.ondemandmapping.org.uk (Fig. 10).

VI. Conclusion and Future Work

The increasing availability of once inaccessible datasets and the explosion of crowd-sourced data, alongside the growth of web-based mapping, have led to the need for on-demand mapping. The requirement to integrate data from a number of disparate sources means that there is a need to automate the creation of the workflow required to generate such maps.

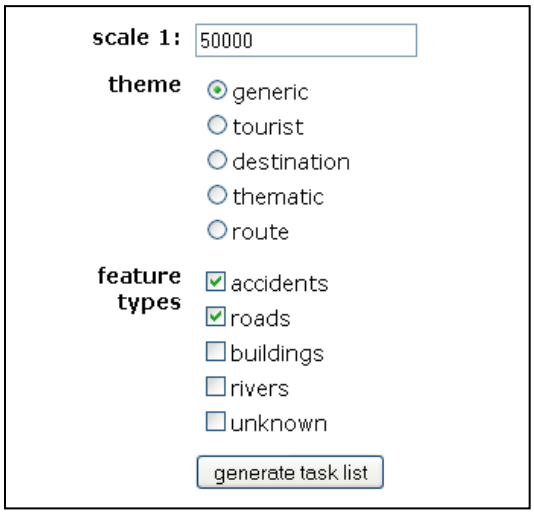

Figure 10. Prompting for user requirements

This paper has presented two aspects of automatic workflows; firstly the generation of the workflow from simple user specifications and secondly the generation of Petri nets from the workflow definitions to allow for their validation. In particular the work done so far has highlighted the potential problem of contradictory rules that can generate deadlocks in workflow definitions.

It was assumed that the generation of the workflow is a static scheduling problem, i.e., the workflow is deterministic, known in advance of execution [22]. This is likely to be a simplification of the on-demand mapping problem; it will be necessary to consider how the workflow may change during execution when, e.g., a particular generalisation service is not available at execution time. For this reason adaptive and autonomic workflow techniques [23][24][25] may need to be investigated. However, it could be argued that any replacement service or set of services would not affect the workflow if the replacement(s) could be represented as a sub-net with a single point of entry and a single point of exit to replace the failed service.

Further work is also required on the means for expressing the cartographic rules. For example, in the case study (Fig. 5) three rules had the same conditions but different actions. Could the rules be expressed more concisely? Also required is further investigation into how the rule base is to be populated and the knowledge hierarchy defined.

The execution of workflows will consist of calling a number of web services that provide generalisation operators. Web services are usually orchestrated using Business Process Execution Language (BPEL) [26]. Once sound workflow nets can be generated and validated using Petri nets it will be useful to investigate the process of generating BPEL from Petri nets [27][28].

Previous research into the orchestration of generalisation services in particular [29][30] will also need to be considered with a view to investigating how to integrate such services into the system.

A major problem with the work done so far is the lack of a data model. The method lacks the concept of tasks doing work on spatial datasets. Datasets have to be managed as they progress through the workflow and conflicts have to be handled when two different tasks attempt to work on the same dataset at the same time. One possibility may be to regard the presence of a dataset as a pre-condition to the firing of a transition. The transition would not fire until the dataset was available. The output from the transition would then be the processed dataset, e.g., a set of clustered accidents.

Whatever the eventual process is employed for generating the workflow, it is believed that the method described in this paper can be used to validate the workflow definition before an execution is attempted.

Acknowledgements

This project is funded by the Dalton Research Institute at Manchester Metropolitan University and the Ordnance Survey of Great Britain. Thanks to Martin Stanton of MMU for guidance on the use Petri nets.

REFERENCES

[1] J. Gaffuri, "Improving web mapping with generalization," Cartographica: The International Journal for Geographic Information and Geovisualization, vol. 46, no. 2, January 2011, pp. 83-91.

[2] E.M. João, Causes and consequences of map generalisation. London: Taylor and Francis, 1998.

[3] Z. Li, "Digital map generalization at the age of enlightenment: A review of the first forty years," Cartographic Journal, vol. 44, no. 1, February 2007, pp. 80-93.

[4] J. Stoter, et al., State-of-the-art of automated generalisation in commercial software. 2010. Available from: http://www.eurosdr.net/projects/generalisation/eurosdr_gen_final_report_mar2010.pdf 01.09.2011

[5] A. Ruas, and C. Duchêne, "A prototype generalisation system based on the multi-agent system paradigm," in Generalisation of Geographic Information, A.M. William, R. Anne, and L.T. Sarjakoski, Eds., Amsterdam: Elsevier Science, 2007.

[6] M. Goodchild, "Citizens as sensors: the world of volunteered geography," GeoJournal, vol. 69, no. 4, 2007, pp. 211-221.

[7] F. Grabler, M. Agrawala, R. W. Sumner and M. Pauly, "Automatic generation of tourist maps," Proc. ACM SIGGRAPH 2008 papers, Los Angeles, California: ACM, 2008.

[8] K. Johannes, A. Maneesh, D. Bargeron, S. David and M. Cohen, "Automatic generation of destination maps," ACM Trans. Graph., vol. 29, no. 6, December 2010, pp. 1-12.

[9] S. Balley and N. Regnauld, "Collaboration for better on-demand mapping," in ICA Workshop on Generalisation, Paris, France, 2011.

[10] N. R. Adam, V. Atluri and W. K. Huang, "Modeling and analysis of workflows using Petri Nets," Journal of Intelligent Information Systems, vol. 10, no. 2, March/April 1998, pp. 131-158.

[11] W. M. P. van der Aalst, "The application of Petri nets to workflow management," Journal of Circuits, Systems and Computers, vol. 8, no. 1 , 1998, pp. 21-66.

[12] A. Aamodt and E. Plaza, "Case-based reasoning: foundational issues, methodological variations, and system approaches," AI Communications, vol. 7, no. 1, 1994, pp. 39-59.

[13] W. M. P. van der Aalst, "On the automatic generation of workflow processes based on product structures," Computers in Industry, vol. 39, no. 2, 1999, pp. 97-111.

[14] S. Chun, V. Atluri and N. Adam, "Domain knowledge-based automatic workflow generation,", Proc. Database and Expert Systems Applications, 13th International Conference, Aix-en-Provence, France, Berlin: Springer, 2002, pp. 778-838.

[15] W. M. P. van der Aalst and K. M. van Hee, Workflow management models, methods, and systems, Cambridge, Mass.: MIT, 2004.

[16] N. Russell, A. ter Hofstede, W. van der Aalst, and N. Mulyar, "Workflow Control-Flow Patterns: A Revised View," Technical Report BPM-06-22, 2006; Available from: http://www.workflowpatterns.com 01.09.2011

[17] T. Murata, "Petri nets: properties, analysis and applications," Proceedings of the IEEE, vol. 77, no. 4, 1989, pp. 541-580.

[18] A. S. Staines, "Rewriting Petri Nets as Directed Graphs," International Journal of Computers, vol. 5, no.2, 2011, pp. 289-297

[19] N. Akharware, PIPE - Platform Independent Petri net Editor 2. 2005; Available from: http://pipe2.sourceforge.net/. 01.09.2011

[20] J. Desel and J. Esparza, Free Choice Petri Nets, Cambridge: Cambridge University Press, 1995.

[21] P. Alimonti, E. Feuerstein, U. Nanni and I. Simon, "Linear time algorithms for liveness and boundedness in conflict-free Petri nets," in LATIN '92 Lecture Notes in Computer Science, Berlin: Springer, 1992, pp. 1-14.

[22] J. W. Herrmann, C.-Y. Lee, and J. L. Snowdon, "A classification of static scheduling problems," in Complexity Issues in Numerical Optimization, P. M. Pardalos, Ed, World Scientific Publishing Co.: Singapore, 1993, pp. 203-253.

[23] R. Muller, U. Greiner, and E. Rahm, "AgentWork: a workflow system supporting rule-based workflow adaptation," Data & Knowledge Engineering, vol. 51, no. 2, 2004, pp. 223-256.

[24] M. Polese, G. Tretola and E. Zimeo. "Self-adaptive management of Web processes," Proc. Web Systems Evolution (WSE), 12th IEEE International Symposium, 2010.

[25] G. Tretola, and E. Zimeo, "Autonomic internet-scale workflows," Proc. of the 3rd International Workshop on Monitoring, Adaptation and Beyond, Ayia Napa, Cyprus, New York: ACM, 2010.

[26] OASIS. Web Services Business Process Execution Language v2.0. 2007; Available from: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf 01.09.2011

[27] P. Sun, C. Jiang and M. Zhou, "Interactive Web service composition based on Petri net," Transactions of the Institute of Measurement and Control, vol. 33, no. 1, 2011, pp. 116-132.

[28] W. M. P. van der Aalst and K.B. Lassen, "Translating unstructured workflow processes to readable BPEL: Theory and implementation," Journal of Information Software Technology, vol. 50, no. 3, 2008, pp. 131-159.

[29] T. Foerster, L. Lehto, T. Sarjakoski, L. T. Sarjakoski, and J. Stoter, "Map generalization and schema transformation of geospatial data combined in a Web Service context," Computers, Environment and Urban Systems, vol. 34, no. 1, 2010, pp. 79-88.

[30] G. Touya, C. Duchêne, and A. Ruas, "Collaborative generalisation: formalisation of generalisation knowledge to orchestrate different cartographic generalisation processes," Proc. of the 6th international conference on Geographic Information Science, Zurich, Berlin: Springer-Verlag, 2010.

[31] D. Dubois, K. Stecke, "Using Petri nets to represent production processes," Proc. of the 22nd IEEE Conference on Decision and Control, 1983.